Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jani Ježˇ

Spletno orodje za primerjavo algoritmov za transkripcijo glasbe

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika dela:

V diplomski nalogi razvijte spletno orodje za primerjavo algoritmov za avtomatsko transkripcijo glasbe. Orodje naj omogoča vključevanje poljubnih algoritmov, dodajanje in izbor zvočnih datotek, nadzor nad poganjanjem algoritmov in podporo za različne formate izhodov. Poleg navedenega naj orodje omogoča še vizualizacijo rezultatov in izračun standardnih evalvacijskih mer ter grafično urejanje in izvoz rezultatov transkripcij v zapisu MIDI.

Mentor: doc. dr. Matija Marolt

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jani Jež sem avtor diplomskega dela z naslovom:

Spletno orodje za primerjavo algoritmov za transkripcijo glasbe

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije Marolta
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 25. avgusta 2016 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Matiji Maroltu, asistentu Matevžu Pesku za strokovno pomoč in odzivnost ter Luki Zakrjašeku za odlično osnovo, na kateri smo zgradili aplikacijo. Poleg staršem in prijateljem pa še posebna zahvala Aniti. Nagyon szépen köszönöm.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
MIR	Music Information Retrival	pridobivanje informacije iz glasbe
MAPS	MIDI Aligned Piano Sounds	poravnani klavirski zvoki MIDI
MIDI	Musical Instrument Digital Interface	glasbeni inštrumentalni digitalni vmesnik
FFT	Fast Fourier transform	hitra Fourierova transformacija
WAV	WAVeform audio format	zvočni format, ki uporablja valovne krivulje
HMM	Hidden Markov Model	skriti model Markova
NMF	Non-negative Matrix Factorization	nenegativna matrična faktorizacija
SVM	Support Vector Machine	metoda podpornih vektorjev
DNMF	Discriminative Non-negative Matrix Factorization	diskriminativna nenegativna matrična faktorizacija
MIREX	Music Information Retrieval Evaluation eXchange	evalvacije sistemov za pridobivanje informacij iz glasbe
API	Application Programming Interface	aplikacijski programski vmesnik
BSD	Berkeley Software Distribution	programska distribucija Berkeley
JSON	JavaScript Object Notation	objektna notacija za JavaScript
CHM	Compositional Hierarchical Model	kompozicionalni hierarhični model

Povzetek

V diplomski nalogi razvijemo spletno orodje, ki omogoča primerjavo algoritmov za samodejno transkripcijo glasbe. Preko grafičnega vmesnika lahko primerjamo časovno zahtevnost izvajanja ter grafično predstavitev in primerjavo rezultatov transkripcij. Nad izbranimi algoritmi omogoča računanje in primerjavo glavnih evalvacijskih mer. Orodje razvijemo na osnovi štirih algoritmov. Na podlagi te vhodne množice aplikacijo posplošimo z možnostjo dodajanja novih. S tem povečamo uporabnost orodja, saj uporabniku omogočimo primerjavo rezultatov lastnih algoritmov s tistimi, ki so v sistem že dodani. Med razvojem aplikacije algoritme izvajamo na zbirki studijskih klavirskih posnetkov MAPS. Ker tovrstne zbirke poleg vsakega zvočnega posnetka vsebujejo še referenčno anotacijo, to izkoristimo ter omogočimo prikaz natančnosti transkripcije posameznega algoritma. Aplikacija omogoča tudi nalaganje novih zvočnih zbirk. Proces samodejne transkripcije glasbe je časovno zahtevnejša operacija, zato dodamo uporabniku možnost, da ob zaključku procesa prejme obvestilo v elektronski poštni predal. Poleg navedenega nam orodje omogoča še grafično urejanje in izvoz rezultata transkripcije v zapisu MIDI. Z omenjenim naborom funkcij na področje raziskovanja samodejne transkripcije glasbe doprinesemo z uporabnim in enostavnim orodjem.

Ključne besede: transkripcija glasbe, pridobivanje informacij iz glasbe, transkripcijski algoritmi, evalvacijske mere, spletna aplikacija

Abstract

This thesis presents the development of a web application that enables the user to compare algorithms for automatic music transcription. The user interface provides visualization of time difficulty, graphic representation and comparison of the algorithm's transcription results. Furthermore, it enables calculation and comparison of main evaluation metrics. The system was developed by using four different transcription algorithms. Based on this input set, the application was generalized by giving the user the ability to add his or her own solutions. This functionality significantly increased usability by providing a comparison of user algorithm results with those already available in the system. For algorithm execution, an annotated sound database, MAPS was used. These sorts of data sets include ground-truth transcriptions for each audio file, therefore adding an extra option to evaluate the algorithm's result accuracy. The application provides a file upload function as well, in order to extend an annotated sound library. Automatic music transcription is generally a time-consuming process, hence an option that notifies the user of its termination had been added. In addition to all functions stated, this tool provides editing and exporting transcription results in MIDI format. With this set of options a simple and useful tool had been introduced to the automatic music transcription community.

Keywords: transcription, music information retrieval, automatic music transcription algorithms, evaluation metrics, web application

Poglavje 1

Uvod

1.1 Transkripcija glasbe

Vsak glasbenik se prej ali slej sreča s problemom učenja in izvajanja glasbenih kompozicij. Tak pristop veliko pripomore k razumevanju glasbe in obvladovanju izbranega inštrumenta. Čeprav imamo ljudje visoke sposobnosti procesiranja v realnem času, bomo po poslušanju zahtevnejše in daljše glasbene kompozicije to v celoti in brez napake s težavo ponovili. Za premagovanje tega problema moramo poslušanja večkrat ponoviti in zaznave zapisati v ustrezno notacijo tako, da jo lahko kasneje rekonstruiramo in zaigramo. Ta postopek se imenuje transkripcija. Proces transkripcije je v primeru enostavnih monofoničnih kompozicij trivialen, saj človeško uho hitro prepozna zaigrani ton. Težave pri transkripciji nastopijo pri glasbenih delih, zaigranih z več različnimi inštrumenti ali s prisotnimi sočasnimi toni (polifonija), morebitnimi šumi na posnetku ali nizkimi legami. Zaradi omenjenih dejavnikov nam postopek transkripcije vzame veliko več časa, kot je sama dolžina kompozicije. Pri tem pa zaradi človeške narave ne izključujemo morebitnih napak in kakovosti rezultata.

Za premagovanje tovrstnih problemov je človek s pomočjo računalnika in matematičnih orodij razvil posebne načine za samodejno transkripcijo glasbe. Tovrstni sistemi so podrobneje opisani v nadeljevnaju. Osnovno idejo pa lahko strnemo v nekaj stavkov. Če zvočni posnetek razstavimo na krajše časovne odseke, lahko vsak del obravnavamo posebej. Za vsak del nato s pomočjo spektralne analize (npr.

hitra Fourierova transformacija ali FFT) ugotovimo, katere frekvence v njem nastopajo. V dobljenem naboru frekvenc nato izločimo tiste, za katere smatramo, da vsebujejo informacijo o zaigranem tonu. Rezultate vseh posameznih delov nato združimo in pretvorimo v človeku razumljiv format.

1.2 Pregled področja

Prvi primer implementacije samodejne transkripcije glasbe zasledimo v sedemdesetih letih. Leta 1977 je James Andy Moorer [1], inženir na področju digitalizacije glasbe, predstavil program za transkripcijo dvoglasnih kompozicij, ki pa je imel svoje omejitve. Da je problem poenostavil in se prilagodil zmogljivosti takratne tehnologije, je izločil vse inštrumente, katerih generirani toni niso celoštevilski večkratniki osnovne frekvence (tolkala, kot so bonbni, činele, gong). Prav tako je predpostavil, da se tonske višine ne spreminjajo. S tem je izključil tehnike igranja, kot so vibrato in glissando. Kot zadnjo omejitev je predpostavil, naj vhodna kompozicija ne vsebuje oktav. V osemdesetih letih raziskave nadaljujejo Piszczalski [12], Maher [3] in Chafe [4]. Do tedaj so bile vse rešitve omejene na maksimalno dvojno stopnjo polifonije. Sledili so poskusi transkripcije kompozicij z višjo stopnjo polifonije, a še vedno omejenih na manjše število prisotnih inštrumentov. Od začetka devedesetih je zanimanje za samodejno transkripcijo glasbe naraslo. Rešitve na osnovi statističnih metod so predstavili Kashino et. al. [5], Goto [6], Ryynanen [7] in Martin [8].

Do danes so inženirji na tem področju razvili in uporabljali vrsto pristopov [28]:

- multimodalna metoda [30]
- nevronske mreže [14]
- skriti model Markova za postprocesiranje (Hidden Markov model - HMM) [7]
- sistem "šolske table"(Blackboard system) [5, 31]
- nenegativna matrična faktorizacija (NMF) [9]
- metoda podprtih vektorjev SVM (Support vector machine) [9]

- avditorni model (Human audition modeling)
- diskriminativna nenegativna matrična faktorizacija (DNMF) [10]
- kombinacija avditornega modela in prilagodljivih oscilatornih mrež [11]
- kompozicionalni hierarhični model [12]

Področje samodejne transkripcije je tudi danes aktivno in težko je oceniti, kam bo razvoj vodil v naslednjih letih [27]. Večina dosedanjih rešitev se zanaša na informacije, ki jih algoritmi pridobijo neposredno iz zvočnih zapisov (data-driven) [25]. Velik doprinos pa se morebiti obeta ob upoštevanju širše množice znanja. Raziskovalci tako v svoje sisteme že skušajo vpeljati znanje iz glasbene teorije in razumevanje človeške percepcije glasbe.

1.3 Motivacija in cilj

Za vrednotenje delovanja in kakovosti vsakega produkta je ključnega pomena, da ga v procesu razvoja primerjamo z že obstoječimi rešitvami na trgu. Na ta način lahko natančneje ocenimo, ali razvoj vodi do cilja, kjer bomo predstavili konkurenčno rešitev. Isto pravilo velja pri razvoju novih algoritmov.

Sistematično testiranje in primerjava delovanja programske kode je dolgotrajen in ponavljajoč proces. Če se lotimo reševanja kompleksnejših problemov, kot je razvoj algoritmov za samodejno transkripcijo glasbe, je potrebno veliko število iteracij, preden lahko iz pridobljenih rezultatov testiranja ocenimo delovanje danega sistema. Vsak tovrsten algoritem namreč definira različno število in tip vhodnih parametrov, kjer vsaka sprememba vrednosti vhodnega parametra spremeni tudi vrednost rezultata na izhodu algoritma. Drugi dejavnik, ki je v procesu testiranja pomemben, je omejitev velikosti testne množice vhodov. Če želimo dani sistem natančno analizirati, moramo na vhod pripeljati čim večji nabor vrendosti. Zanima nas namreč obnašanje sistema tako v najbolj pogostih situacijah kot tudi v robnih primerih. Poleg tega sam algoritem med procesom razvoja konstantno spreminjamo in izboljšujemo ter s tem vplivamo na rezultat. Število iteracij testiranja tako že pri majhnem naboru vhodnih parametrov naraste in rezultira v ogromno količino izhodnih podatkov. Ti podatki so ponavadi v obliki, ki nam ne daje jasnih

odgovorov, in jih je zato potrebno dodatno obdelati. Obdelave rezultatov se lahko lotimo s pomočjo obstoječih orodij ali pa sami razvijemo sisteme, ki znajo iz pridobljenih podatkov bistvo izluščiti in ustrezno prikazati. Na področju samodejne transkripcije glasbe sicer že obstajajo orodja, ki omogočajo grafično vizualizacijo rezultatov. Primer takih orodij so npr. komercialne rešitve Transcribe! ali Capo 3. Ta orodja so osredotočena na končne uporabnike, ki jih zanima le rezultat transkripcije. Za raziskovalca na tem področju pa še vedno nimamo rešitve, ki bi na enem mestu združevala prikaz transkripcij in vrednotenje natančnosti rezultatatov algoritmov.

Če želimo realizirati uporabno evalvacijsko orodje, je potrebno problematiko najprej ustrezno raziskati. Vključiti moramo vse kriterije, ki so na danem področju relevantni. V primeru transkripcijskih sistemov se je pomembno osredotočiti na meritve časovne zahtevnosti izvajanja algoritmov, natančnost rezultatov transkripcije in omogočanje primerjave rezultatov z ostalimi algoritmi. Pomemben kriterij za primerjavo so tudi evalvacijske mere, ki so natančno definirane v sklopu nabora pravil MIREX (Music Information Retrieval Evaluation eXchange), ki je nastalo pod okriljem laboratorija IMIRSEL (International Music Information Retrieval Systems Evaluation Laboratory) v Illinoisu.

Motivacija in končni cilj naloge je razviti manjkajoči sistem, ki bo razvijalcem algoritmov za samodejno transkripcijo glasbe na podlagi vseh omenjenih kriterijev omogočil podroben pregled in primerjavo rezultatov njihovega dela z obstoječimi sistemi.

Poglavje 2

Transkripcijski algoritmi

Cilj transkripcijskih algoritmov je zapis glasbene kompozicije v človeku berljiv format s pomočjo računalnika. Problem, ki ga tovrstni sistemi rešujejo, lahko razdelimo na naslednje podprobleme:

- detekcija več hkratnih tonskih višin
- detekcija začetka in konca tona
- detekcija glasnosti tona
- razločevanje različnih glasbenih inštrumentov in šumov
- pridobivanje informacije o tempu in ritmiki
- prikaz pridobljenih rezultatov v berljivem formatu

Potreba po reševanju vseh omenjenih nalog je odvisna od zvrsti glasbe in zahtev s strani končnega uporabnika. Glavni od naštetih podproblemov je detekcija osnovnih tonskih višin v danem časovnem odseku.

Glede na pristop k implementaciji in predstavitvi rezultatov na izhodu, lahko algoritme razdelimo v tri skupine:

- detekcija tonov v časovih odsekih (frame-level)
- detekcija začetka in trajanja tonov (note-level)
- sledenje barvi tona (stream-level)

Poleg pristopa k implementaciji algoritme medseboj razlikuje tip in število vhodnih parametrov, format izhoda in platforma, na kateri jih je možno izvajati. V procesu razvoja aplikacije smo za osnovo uporabili štiri transkripcijske algoritme, ki zajamejo nekaj omenjenih razlik.

2.1 Benetos

Platforma: Matlab / octave
Format izhoda: klavirska tablatura
Pristop k implementaciji: frame-level
Opis delovanja:
Leta 2015 Benetos [13, 24] predstavi sistem, ki omogoča učinkovito transkripcijo več inštrumetnalnih kompozicij. V raziskavi preizkusi dva modela. Prvi deluje na osnovi analize skritih komponent (probabilistic latent component analysis - PLCA). PLCA je način frekvenčne razčlenitve, ki se poleg nenegative matrične faktorizacije (NMF) pogosto uporablja v transkripcijskih sistemih. Drugi, časovno omejeni model (temporally-constrained model), ponazarja evolucijo vsakega tona kot zaporedje zvočnih stanj. Slednji se izkaže kot boljši. Kot vhod obeh metod uporabi časovno frekvenčno reprezentacijo posnteka, ko jo pridobi s pomočjo variabilne Q transformacije (VQT). Z rešitvijo pokaže, da uporaba VQT v primerjavi z bolj razširjeno konstantno Q transformacijo (CQT) doseže boljše rezultate transkripcij.

2.2 SONIC

Platforma: sistemska konzola Windows okolja
Format izhoda: MIDI, MIREX F0, MIREX Note
Pristop k implementaciji: note-level & frame-level
Opis delovanja:
Sistem je leta 2004 predstavil Marolt [14, 11]. Za razliko od večine takratnih rešitev pod drobnogled postavi podproblem sledenja delov (partials tracking). Del je ekvivalent frekvenčni komponenti, vsota teh pa v grobem tvori ton glasbenega inštrumenta. Metoda temelji na kombinaciji avditornega modela in prilagodlji-

vih oscilatornih mrež. Naloga prvega koraka je transformacija zvočnega zapisa na vhodu v časovno frekvenčni prostor, ki je osnova za drugi korak. Naloga drugega koraka je sledenje skupin harmonično povezanih delov z združevanjem prilagodljivih oscilatorjev v mreže. Omenjeni pristop nato uporabi v sistemu za transkripcijo glasbe - SONIC. S sistemom predstavi izboljšave natančnosti rezultatov transkripcije. V SONIC dodatno vključi še detekcijo začetka in ponavljanja ter zaznavo dolžine in glasnosti tonov.

2.3 CHM

Platforma: sistemska konzola Windows okolja
Format izhoda: klavirska tablatura
Pristop k implementaciji: frame-level
Opis delovanja:
Leta 2014 Pesek et. al [12] predstavi kompozicionalni hierarhični model. Zaradi robustne narave modela lahko z njim rešujemo različne probleme na področju pridobivanja informacije iz glasbe. Uporabimo ga lahko za evalvacijo akordov, vizualizacijo in sintezo zvoka. Model poleg tega daje kakovostno informacijo o tonskih višinah, zato ga lahko uporabimo kot sistem za transkripcijo glasbe.

2.4 Klapuri

Platforma: Matlab skripta
Format izhoda: matrika F0 in istoležna matrika navzčnosti frekvenc
Pristop k implementaciji: frame-level
Opis delovanja:
Leta 2006 Klapuri [15] predstavi sistem za detekcijo osnovnih frekvenc s pomočjo seštevanja harmonskih amplitud. Da poveča robustnost sistema, spekter zvočnega signala na vhodu zgladi in s tem oslabi neželjene informacije. Vsako od osnovnih frekvenc (F0) definira z vrednostjo, ki predstavlja njeno navzočnost (salience). Izračuna jo kot uteženo vsoto amplitud harmoničnih delov dane F0. V raziskavi predstavi tri metode za izračun navzočnosti. Direktna metoda izračuna navzočnosti F0 v danem časovnem okviru ter izloči željeno število lokalnih ma-

ksimumov. V drugi metodi uporabi maksimalno navzočnost v danem časovnem okviru. Frekvenco, ki temu maksimumu pripada, nato pri detekciji vsake nasledenje izloči. Tretja metoda skuša detektirati vse osnovne frekvence hkrati. Za rezultat na izhodu algoritma dobimo matriko osnovnih frekvenc in matriko navzočnosti za vsak časovni odsek. Algoritem, ki smo ga uporabili med razvojem orodja, je implementiran z iterativno metodo.

2.5 Primeri izhodov transkripcijskih algoritmov

V nadeljevanju so prikazani formati najpogostejših izhodov transkripcijskih algoritmov.

2.5.1 MIDI

MIDI (Musical Instrument Digital Interface) je protokol, razvit v osemdesetih letih in določa standard za medsebojno komunikacijo digitalnih glasbenih naprav. V zapisu MIDI lahko med drugim preberemo začetek in trajanje tonov, njihovo višino in glasnost. S pomočjo nabora orodij Toolbox util 1.1 [17] lahko iz MIDI datotek razberemo podatke v obliki 7 stolpične matrike v naslednjem zaporedju:

- začetek tona v udarcih
- trajanje tona v udarcih
- MIDI kanal
- višina tona
- glasnost
- začetek tona v sekundah
- trajanje tona v sekundah

2.5.2 Klavirska tablatura

Rezultat transkripcije opisuje dvodimenzionalna matrika (2.5.2). Vsak izmed 88 stolpcev matrike predstavlja en ton na klavirski tipkovnici, ki pokriva razpon od 27 Hz ali ton A0 do 4186 Hz ali ton C8. Vrstice matrike predstavljajo odseke vhodnega zvočnega posnetka in jih je lahko poljubno mnogo. Dolžina enega odseka je določena z implementacijo algoritma. Najpogosteje je dolžina časovnega odseka 10ms. Vrednost vsake celice je navzočnost tona za dani časovni odsek.

$$M = \begin{bmatrix} x_{1,1} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{2,1} & x_{22} & x_{23} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ x_{88,1} & x_{d2} & x_{d3} & \dots & x_{dn} \end{bmatrix} \quad (2.5.2)$$

2.5.3 Istoležna matrika osnovnih frekvenc in njihovih navzočnosti

Na izhodu algoritma dobimo dve dvodimenzionalni matriki (2.5.3). Prva določa vrednosti detektiranih osnovnih frekvenc, druga pa vrednost navzočnosti detektiranih F0 v danem časovnem odseku. Dimenzije matrike so spremenljive in določene z nastavitvijo vhodnega parametra algoritma ter pogojene z dolžino zvočnega zapisa in omejitvijo časovnega odseka.

$$F0 = \begin{bmatrix} 3996.70 & 4392.94 & 4281.12 & \dots & 4406.34 \\ 80.72 & 78.91 & 320.11 & \dots & 242.197 \\ \dots & \dots & \dots & \dots & \dots \\ 812.90 & 815.72 & 786.31 & \dots & 655.31 \end{bmatrix}$$

(2.5.3)

$$Sals = \begin{bmatrix} 11.2 & 12.6 & 12.4 & \dots & 11.4 \\ 8.4 & 6.3 & x_{23} & \dots & 7.3 \\ \dots & \dots & \dots & \dots & \dots \\ 1.2 & 1.1 & 3.2 & \dots & 1.8 \end{bmatrix}$$

2.5.4 MIREX F0

Format opisuje tabela vrednosti (2.5.4), kjer vsaka vrstica ponazarja en časovni odsek (frame). V vsaki vrstici prvi stolpec ponazarja časovni odmik od začetka zvočnega zapisa, vsi naslednji pa vsebujejo vrednosti osnovnih frekvenc, detektiranih v danem odseku.

časovni odmik	F01	F02	...	F0n
0.45	145.22	235.00		
0.46	653.23	1760.23	1778.34	...
0.47	...	...		

(2.5.4)

2.5.5 MIREX Note

Format opisuje tabela vrednosti (2.5.5), kjer vsaka vrstica vsebuje informacijo o začetku in trajanju detektirane osnovne frekvence. Vrstice so urejene glede na začetek pojavitve osnovne frekvence F0. Prvi podatek je začetek pojava frekvence, drugi njeno trajanje, zadnji pa vrednost osnovne frekvence.

začetek	trajanje	F0
0.45	1.5	27.00
0.64	3.5	1760.23
0.64	6	234.54
...	...	...

(2.5.5)

2.6 Zbirke testnih zvočnih datotek

Pri razvoju sistemov za transkripcijo so ključnega pomena zbirke tesnih datotek, nad katerimi algoritme izvajamo. Če hočemo točnost algoritma ustrezno ovrednotiti, ga je potrebno primerjati z referenčimi anotacijami. Tovrstne informacije lahko pridobivamo avtomatsko s pomočjo programov kot so Disklavier (samo za klavirsko glasbo), polavtomatsko, tako da ročno popravimo rezultate na izhodu sistemov za detekcijo osnovnih frekvenc, ter ročno v primeru kompleksnejših kompozicij. Pri iskanju zvočnih zbirk najpogosteje naletimo na transkripcije solo kla-

virske glasbe. Za ostale inštrumente so zbirke še vedno manj dostopne. Da lahko rezultate danega algoritma smatramo za reprezentativne, moramo transkripcije izvajati na večji množici vhodnih datotek, drugače lahko zmožnosti sistema precenimo.

Med razvojem primerjalnega orodja smo uprabili zbirko zvočnih datotek MAPS. Ta nam ponuja približno 65 ur zvočnih posnetkov v formatu WAV. Posneti so v CD kakovosti (16bit, 44kHz stereo). Poleg vsakega posnetka so dodane referenčne transkripcije v obliki poravnane MIDI datoteke in tekstovne datoteke, ki za vsak ton vsebuje začetek, trajanje in njegovo višino. Na posnetkih je uporabljenih več različnih tipov klavirjev in snemalnih razmer.

Poleg omenjene zbirke omenimo še nekaj ostalih:

- UIOWA Musical Instrument Samples - strunski inštrumenti, pihala in tolkala
- RWC Musical Instrument Sounds - zvočni posnetki 50 različnih inštrumentov
- McGill University Master Samples - orkestralni inštrumenti
- TRIOS dataset - posnetki in referenčne datoteke jazz trio klasikov

Poglavje 3

Evalvacijske mere

Za evalvacijo transkripcijskih sistemov se pogosto uporabljajo mere, ki so v uporabi tudi pri vsakoletni evalvaciji algoritmov pridobivanja informacij iz glasbe MIREX (Music Information Retrieval Evaluation eXchange) [40, 41]. MIREX združuje nabor formalnih evalvacij, določenih s strani znanstvene sfere, ki se ukvarja s pridobivanjem informacije iz glasbe (music information retrival - MIR). Nadzorovan je s strani mednarodnega laboratorija za evalvacijo sistemov za pridobivanje informacij iz glasbe (International Music Information Retrieval Systems Evaluation Laboratory - IMIRSEL) na univerzi v Illinoisu. Za evalvacijo MIREX se uprabljajo referenčne transkripcije, pridobljene s pomočjo posebnih sistemov. Ti detektirajo tone v monofoničnih izvedbah originalov, posnetih v nadzorovanem okolju. Rezultati so nato še ročno pregledani in popravljeni.

Kot smo že omenili v prejšnjem poglavju, na izhodu transkripcijskih algoritmov pričakujemo rezultat, ki je za človeka razumljiv in uporaben. Tovrstni sistemi do končnega zapisa transkripcije pridejo preko vmesnega koraka. To je zapis vseh osnovnih frekvenc za vsak časovni odsek (v primeru frame-level pristopa), torej rezultat procesa detekcije osnovnih frekvenc. Tak zapis vsebuje informacije, ki jih lahko poleg transkripcije uprabimo tudi v druge namene. (npr. za pomoč sistemom za avtomatsko iskanje glasbe).

En od načinov za uporabo rezultata detekcije F0 je evalvacija natančnosti algoritmov. Glede na metodologijo evalvacije razlikujemo:

- vrednotenje več osnovnih frekvenc (MIREX multiple F0)

- vrednotenje na osnovi sledenja tonov (note tracking)

Pogosto se prvi pristop vrednotenja uprabi kot predhodnik drugega. Vsaka od omenjenih metodologij definira več evalvacijskih mer. Glavne tri so:

- natančnost detektiranih F0 v danem časovnem odseku (Multiple F0 precision) (3.1)
- število pravilno detektiranih F0 v danem časovnem odseku (Multiple F0 recall) (3.2)
- povprečje obeh (Multiple F measure) (3.3)

3.1 Evalvacija MIREX multi F0

Metodologija primerja rezultat za vsak časovni odsek posebej. Dolžina časovnega odseka je 10ms.

Izračun glavnih evalvacij definirajo naslednje enačbe:

$$Precision = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t)} \tag{3.1}$$

$$Recall == \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FN(t)} \tag{3.2}$$

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{3.3}$$

Spremenljivke v enačbah definiramo kot:

- TP - število pravilno detektiranih F0 v časovnem odseku t (true positives)
- FP - število detektiranih F0, ki v referenčni transkripciji na časovnem odseku t ne obstajajo (false positives)
- FN - število F0, ki v referenčni transkripciji obstajajo, a jih sistem za dani časovni odsek t ni detektiral (false negatives)

Število virov v vsakem časovnem odseku je spremenljivo. To vpliva na obseg detektiranih osnovnih frekvenc. Zaradi tega so spremenljivke TP(t), FP(t) in FN(t) definirane kot funkcija v odvisnosti od časa t.

Če vrednosti spremenljivk TP(t) FP(t) in FN(t) za vsak časovni odsek seštejemo, dobimo *natančnost (accuracy)* rezultata analize za celotno kompozicijo:

$$Accuracy = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t) + FN(t)} \tag{3.4}$$

Vrednost, ki jo dobimo kot rezultat, je v obsegu med 0 in 1, kjer 1 ponazarja popolno enakost rezultata na izhodu sistema z referenčno transkripcijo.

Pogosto se zgodi, da določen vir (glasbeni inštrument) v kompoziciji nastopa le kratek čas. Med procesom detekcije F0 se lahko zgodi, da algoritem nepravilno oceni vir detektirane F0, pomotoma vstavi ali pa v celoti izpusti detekcijo F0 določenega vira. S pomočjo do sedaj opisanih meritev pa ne dobimo podatka o omenjenih vzrokih napak, ki jih je sistem napravil. MIREX zato vključuje oceno transkripcijske napake za posamezen časovni odsek (frame-level transcription error score). Izračun *skupne napake* definira naslednja enačba:

$$E_{tot} = \frac{\sum_{t=1}^{T} max(N_{ref}(t), N_{sys}(t)) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)} \tag{3.5}$$

Spremenljivke v enačbi definiramo kot:

- $N_{ref}(t)$ - število F0 v referenčni datoteki v časovnem odseku t
- $N_{sys}(t)$ - število detektiranih F0 v časovnem odseku t
- $N_{corr}(t)$ - število pravilno detektiranih F0 v časovnem odseku t

S pomočjo te enačbe lahko določimo posamezne tipe napak. Izračunamo lahko zamenjavo (substitution), zgrešitev (miss) in lažni alarm (false alarm).

Zamenjava nam poda procent referenčnih F0, ki jih sistem ni detektiral ter namesto njih vrnil napačne vrednosti. Definira jo enačba (3.6):

$$E_{sub} = \frac{\sum_{t=1}^{T} min(N_{ref}(t), N_{sys}(t)) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)} \tag{3.6}$$

Zgrešitev nam poda procent referenčnih F0, ki jih je sistem izpustil in namesto njih ni vrnil nobene vrednosti. Definira jo enačba (3.7):

$$E_{miss} = \frac{\sum_{t=1}^{T} max(0, N_{ref}(t) - N_{sys}(t))}{\sum_{t=1}^{T} N_{ref}(t)} \tag{3.7}$$

Lažni alarm nam poda procent detektiranih F0, ki niso smatrane kot zamenjava. Definira ga enačba (3.8):

$$E_{fa} = \frac{\sum_{t=1}^{T} max(0, N_{sys}(t) - N_{ref}(t))}{\sum_{t=1}^{T} N_{ref}(t)} \tag{3.8}$$

3.2 Evalvacija MIREX note

Pri algoritmih, ki uporabljajo metodologijo sledenja tonov, je pomembno ocenjevanje točnosti začetka in trajanja ter prekrivanja detektiranih F0. Referenčna datoteka vsebuje seznam dogodkov. Vsak dogodek je definiran s tremi vrednostmi. To so, začetek pojavitve tona, trajanje tona ter njegova frekvenčna vrednost.

Ker je generiranje tovrstnih referenčnih podatkov v kompleksnejših polifoničnih kompozicijah zahtevnejše, evalvacija dopušča določen prag napake. Privzamemo lahko, da se začetek tona smatra za pravilen z dopuščanjem napake do +/- 50 milisekund. Detekcija F0 se smatra za pravilno, če se od referenčne vrednosti razlikuje za do 3% (četrt tona). Pri vrednotenju trajanja tona se za pravilno detektirano vrednost upošteva do 20% razlikovanje (a najmanj 50 milisekund) z referenčno dolžino. Na podlagi teh predpostavk se nato s pomočjo enačb (3.1), (3.2), (3.3) izračunajo osnovne evalvacijske mere *precision*, *recall* in $f-measure$.

Pri MIREX note evalvaciji spremenljivke v enačbah definiramo kot:

- TP - število detektiranih tonov, ki ustrezajo danim predpostavkam (true positives)
- FP - število detektiranih tonov, ki ne ustrezajo danim predpostavkam (false positives)
- FN - število tonov, ki v referenčni transkripciji obstajajo, a jih sistem ni detektiral (false negatives)

S pomočjo enačbe (3.4) lahko ocenimo še natančnost rezultata analize za celotno kompozicijo (Accuracy).

Poleg omenjenih evalvacij se pri tej metodologiji ocenjuje tudi stopnja prekrivanja tonov (Overlap Ratio). Računamo jo s pomočjo enačbe (3.9):

$$OR_i = \frac{min(t^{ref}_{i,off}, t^{sys}_{i,off}) - max(t^{ref}_{i,on}, t^{sys}_{i,on})}{max(t^{ref}_{i,off}, t^{sys}_{i,off}) - min(t^{ref}_{i,on}, t^{sys}_{i,on})} \tag{3.9}$$

Spremenljivke v zgornji enačbi definiramo kot:

- OR_i - stopnja prekrivanja za i-ti pravilno detektiran ton
- $t^{ref}_{i,off}$ - referenčni čas trajanja tona
- $t^{ref}_{i,on}$ - referenčni čas začetka tona
- $t^{sys}_{i,off}$ - detektirani čas trajanja tona
- $t^{sys}_{i,on}$ - detektirani čas začetka tona

3.3 Oktavno invariantne evalvacije

Pri procesu detekcije osnovnih frekvenc se veliko napak nanaša tudi na napačno zaznavanje harmonikov oz. večkratnikov osnovne frekvence (oktav). Zaradi tega se pri vrednotenju transkripcijskih sistemov uporablja še nabor evalvacij, ki oktavne napake ignorirajo (Chroma accuracy). V tem načinu se pri računanju omenjenih osnovnih kriterijev vse detektirane oktave za dano F0 združi v eno samo. To je seveda potrebno narediti tudi za vrednosti v referenčni datoteki. Evalvacijske enačbe ostanejo nespremenjene. Ta kriterij se računa za obe omenjeni metodologiji (MIREX Note, MIREX Multiple F0).

Poglavje 4

Spletno orodje

4.1 Splošno

Za primerjavo in vizualizacijo rezultatov transkripcijskih algoritmov smo razvili spletno aplikacijo, ki zajema vse funkcionalnosti, pričakovane v tovrstnih sistemih. Enostaven uporabniški vmesnik nam omogoča upravljanje z algoritmi ter zbirkami testnih datotek, nad katerimi transkripcije izvajamo. Osrednji del aplikacije je namenjen konfiguraciji procesa analize in pregledu rezultatov. Poleg tega imamo na voljo še zbirko pripomočkov, s katerimi transkripcije urejamo in izvozimo za uporabo v drugih orodjih.

4.2 Uporabljena orodja in tehnologije

Arihtektura aplikacije je razdeljena na tri dele:

- aplikacija na strani odjemalca
- aplikacija na strani strežnika
- upravljanje s podatki

Sistem (Slika 4.1) je natančneje opisan v nadeljevanju.

4.2.1 Implementacija na strani odjemalca

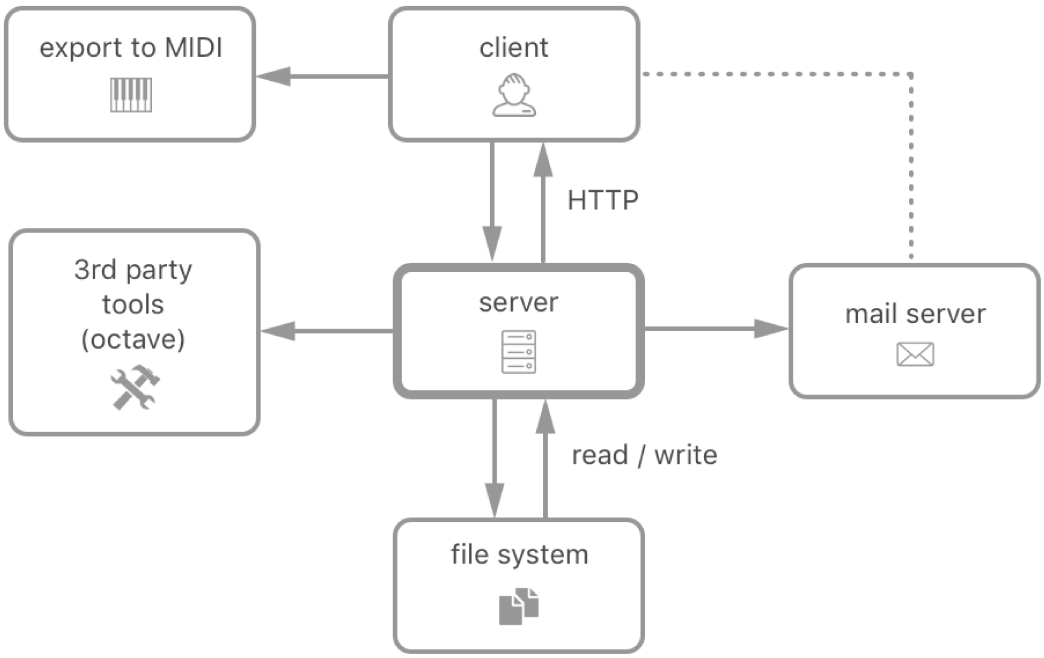
Ta del aplikacije se izvaja v brskalniku na strani odjemalca. V osnovi je zgrajen s pomočjo spletnih tehnologij kot so JavaScript, HTML5 in CSS3. Za razvoj spletnih rešitev imamo danes na voljo veliko zbirko orodij, ki razvijalcem olajšajo delo. V našem primeru smo pri implementaciji JavaScript logike uporabili ogrodje AngularJS. Ta ponuja bogat nabor rešitev, ki jih razvijalcu ni potrebno ponovno implemetirati. Ta del aplikacije med drugim skrbi za komunikacijo s spletno rešitvijo preko protokola HTTP, obdelavo in zapis pridobljenih podatkov v obliko statičnih HTML dokumentov, ki jih nato brskalnik na strani odjemalca prikazuje. Za urejanje prikaza tovrstnih dokumentov smo uporabili programski jezik SCSS. Z njim definiramo prezentacijo HTML dokumentov in s tem izboljšamo uporabniško izkušnjo. Da smo si ta del naloge poenostavili, smo uporabili ogrodje Bootstrap.

4.2.2 Implementacija na strani strežnika

Del aplikacije, ki se izvaja na strani strežnika, skrbi za obdelavo in serviranje zahtev odjemalca. Z vidika odjemalca je definirana kot aplikacijski programski vmesnik (API), ki temelji na arhitekturnem pristopu REST. Ta pristop je prvič opisal Roy Fielding [20]. Njegov namen je opis naslavljanja in definicije virov, do katerih dostopamo npr. s pomočjo protokola HTTP. Informacije o virih definira ponudnik spletne storitve. Na ta način omogočimo odjemalcu svobodo glede izbire načina obdelave podatkov. Logika na strani strežnika je napisana v programskem jeziku C#. Vsebine virov, ki se izmenjujejo med odjemalcem in strežnikom, so zapisane v formatu JSON. Za implementacijo smo uporabili odprtokodno platformo MONO, ki bazira na ogrodju .NET. Na ta način programerjem, vključenim v projekt, omogočimo svobodo pri izbiri razvojnega okolja kot so Windows, Linux, BSD in OS X.

4.2.3 Podatki

Za zapis in shranjevanje konfiguracije dodanih algoritmov ter rezultate transkripcij sistem uporablja statične datoteke. Za vsak proces se tako na disku ustvari direktorij, v katerega ločeno shranjuje rezultate za posamezen algoritem in zvočno



Slika 4.1: Slika glavnih relacij med deli sistema

datoteko. Konfiguracija algoritmov je shranjena v skupni datoteki, v formatu JSON.

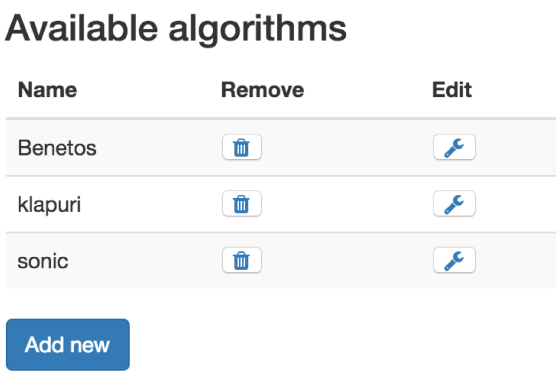
4.3 Sestavni deli in opis delovanja

Uporabniški vmesnik aplikacije je razdeljen na tri dele:

- sistem za upravljanje z algoritmi
- sistem za upravljanje z datotekami
- proces konfiguracije analize in pregled rezultatov

4.3.1 Upravljanje z algoritmi

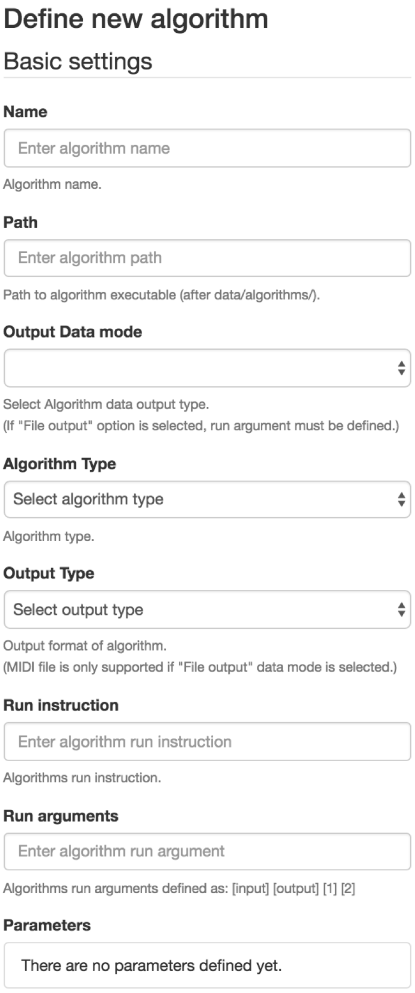
Z namenom, da uporabnost aplikacije čim bolj posplošimo in približamo širši množici razvijalcev transkciprijskih sistemov, smo v nabor funkcij dodali možnost



Slika 4.2: Seznam obstoječih algoritmov

definiranja lastnih algoritmov. Na ta način lahko uporabnik preko grafičnega vmesnika definira ukaz in vhodne parametre, ki jih zahteva podpis algoritma. Poleg omenjene funkcionalnosti imamo na tem mestu na voljo še pregled nad vsemi že obstoječimi algoritmi ter možnost urejanja njihovih parametrov. Grafični vmesnik smo razdelili na tri dele:

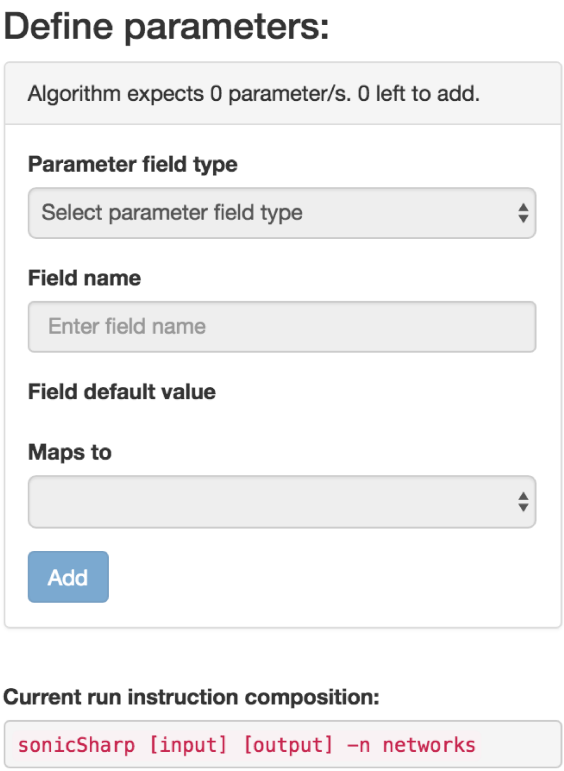
- seznam vseh obstoječih algoritmov (Slika 4.2)
- obrazec za dodajanje ali urejanje algoritma (Slika 4.3)
- obrazec za definiranje in dodajanje parametrov ukaza (Slika 4.4)



Slika 4.3: Obrazec za dodajanje in urejanje algoritmov

Seznam algoritmov nam za vsak element omogoča dve operaciji. Izbrani algoritem lahko odstranimo iz sistema ali pa urejamo njegove parametre. Vsak algoritem je določen z spodnjimi vrednostmi:

- ime algoritma
- pot do ukaza za izvšitev algoritma
- način branja podatkov na izhodu algoritma



Slika 4.4: Obrazec za definiranje parametrov

- vrsta algoritma
- format izhodnih podatkov
- ime ukaza, s katerim algoritem poženemo
- zaporedje in tip vhodnih parametrov
- seznam vseh parametrov s privzetimi vrednostmi

Ime algoritma

S tem parametrom določimo ime algoritma, ki je prikazano preko celotne aplikacije.

Pot do ukaza za izvršitev algoritma

V primeru, da je datotečna zgradba uporabnikovega algoritma večnivojska, s tem parametrom določimo pot do izvršljive datoteke. Vsi algoritmi se v sistemu nahajajo na istem naslovu, zato je absolutna pot do korenskega direktorija, ki algoritme vsebuje, že določena. Uporabnik mora definirati le pot od korenskega direktorija dalje.

Način branja podatkov na izhodu algoritma

Na osnovi štirih testnih algoritmov smo podprli dva načina branja podatkov, ki ju lahko pričakujemo na izhodu. Algoritem lahko končni rezultat transkripcije zapiše:

- v izhodno datoteko, ki jo nato sistem prebere
- v obliki toka podatkov, ki ga sistem med izvajanjem algoritma posluša in bere iz sistemske konzole

Vrsta algoritma

Da bi uporabnikom omogočili čim večjo svobodo pri izbiri platforme za razvoj algoritmov, smo vgradili podporo za Matlab okolje in izvršljive datoteke EXE za okolje Windows. S tem parametrom uporabnik izbere željeno vrednost, na osnovi katere sistem v ozadju sestavi ukaz in parametre, ki so potrebni za zagon programa.

S tem naborom možnosti omogočimo, da lahko uporabnik svoje algoritme razvija na vseh bolj razširjenih operacijskih sistemih, ki so danes na voljo.

Format izhodnih podatkov

Med integracijo testnih algoritmov smo opazili, da ima vsak od njih na izhodu lahko drugačen format rezultata. Formati, s katerimi smo se srečali so:

- klavirska tablatura (piano roll)
- izhodna datoteka z zapisom MIDI
- MIREX F0 format
- MIREX Note format
- Istoležna matrika osnovnih frekvenc in njihovih navzočnosti

Ker so omenjeni izhodi za tovrstne sisteme najbolj tipični, smo v aplikacijo vgradili podporo za vsakega od njih. Le v primeru zadnjega na seznamu smo izhod s pomočjo dodatne skripte prestregli in ga predelali v obliko klavirske tablature. Zaradi modularnosti sistema bi bila nadgradnja s podporo za ta format trivialna.

Ime ukaza za zagon algoritma

Tu določimo ukaz za zagon algoritma. V primeru Matlab skripte je to kar ime funkcije, v primeru datoteke EXE pa ime programa. Format ukaza mora ustrezati zahtevam izbrane vrste algoritma.

Zaporedje in tip vhodnih parametrov

Tu določamo zaporedje vhodnih parametrov danega algoritma. To zaporedje se nato uporabi pri sestavljanju končnega ukaza, ki ga bo sistem uporabil za zagon. Pri navedbi parametrov razlikujemo med vhodno in izhodno datoteko ter ostalimi paramteri, ki so definirani z zaporednimi števili, začenši z 1. Razlog za ločevanje tipov vhodnih parametrov je omogočiti zagon procesa transkripcije nad več datotekami za isti algoritem. Posledica te zahteve je, da se tako vhodna kot izhodna datoteka med samim izvajanjem analize lahko spreminajata. Z vpeljavo razlikovanja tipa vhodnih parametrov enostavno dosežemo to funkcionalnost.

Seznam vseh parametrov z privzetimi vrednostmi

V tem delu uporabniškega vmesnika je prikazan rezultat postopka definicije vhodnih paramerov za dani algoritem. Tu lahko poljubno urejamo privzete vrednosti posameznih parametrov ali pa jih v celoti odstranimo. Postopek konfiguracije je podrobneje opisan v nadaljevanu.

Konfiguracija vhodnih parametrov algoritma

Ko smo določili število in zaporedje vhodnih parametrov, lahko vsakega od njih natančneje definiramo. Vhodni parameter algoritma definirajo tri lastnosti:

- tip
- ime
- privzeta vrednost

S pomočjo imena razlikujemo med vhodnimi parametri. S tipom parametra določimo nabor vrednosti, ki jih ta zahteva. Aplikacija podpira tri tipe. To so, številčna vrednost, Boolova vrednost (true ali false) in znakovni niz. Na ta način podpremo vse možnosti, ki se lahko pojavijo na vhodu danega algoritma. Dodatno lahko za vsak parameter določimo privzeto vrednost, ki bo uporabljena v procesu analize, v kolikor je uporabnik med postopkom konfiguracije ne spreminja.

Ker je pri definiciji vhodnih parametrov pomembno tudi njihovo zaporedje, smo v ta postopek vgradili dodatno izbirno polje, preko katerega uporabnik določa pozicijo parametra v ukazu (mapping).

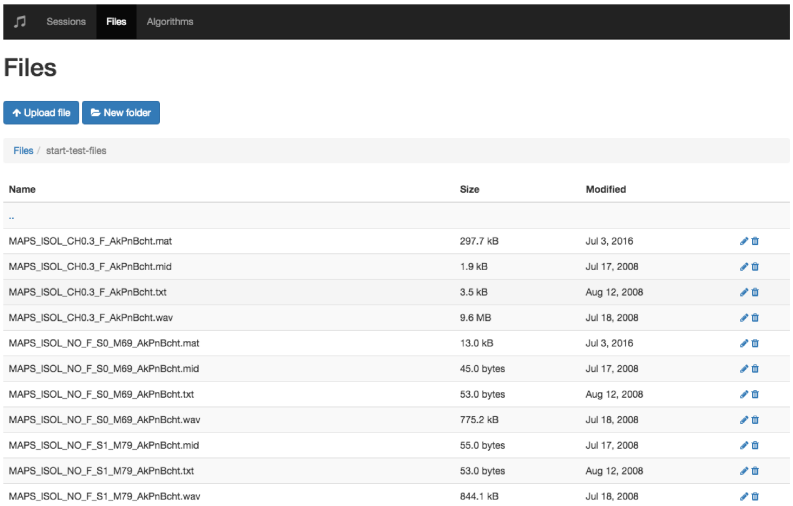
Da dosežemo jasen pregled nad procesom konfiguracije podpisa algoritma, je uporabniku v pomoč predogled ukaza, ki se posodablja med samim urejanjem.

Zaključek konfiguracije

Ko smo smo določili vse naštete parametre, lahko algoritem shranimo v sistem. To storimo z izbiro gumba Submit. Ta akcija poskrbi, da se definicija algoritma shrani na strežniku v posebno datoteko, ki hrani tovrstne zapise, v obliki formata JSON. To pomeni, da smo v sistem dodali nov algoritem, ki ga sedaj lahko izvajamo in primerjamo z ostalimi, ki so že na voljo.

4.3.2 Upravljanje z zbirkami datotek

V tem delu aplikacije omogočamo nalaganje in urejanje zbirke zvočnih datotek, ki jih kasneje uporabljamo kot osnovo za izvajanje algoritmov. Vmesnik (Slika 4.5) omogoča pregled datotek, njihovo odstranjevanje ali spremembo imena. Za vsako od njih je navedena osnovna informacija o velikosti na disku ter datum zadnje spremembe. Če želimo dodatno urejati datotečno strukturo, lahko to storimo z dodajanjem direktorijev. V prvem koraku konfiguracije analize je komponenta za kontrolo nad datotekami prilagojena s funkcijo izbire posameznih datotek in podatka o dolžini zvočnega zapisa. Poleg tega v tem delu prikazujemo le datoteke formata WAV, ker je to edini format, ki ga za izvajanje algoritmov orodje podpira.



Slika 4.5: Grafični vmesnik za urejanje in dodajanje zbirk datotek

4.3.3 Konfiguracija in pregled procesov

S pomočjo kreiranja novih sej lahko v sistemu izvajamo poljubno število procesov transkripcije. Proces je potrebno najprej definirati. To storimo v treh korakih:

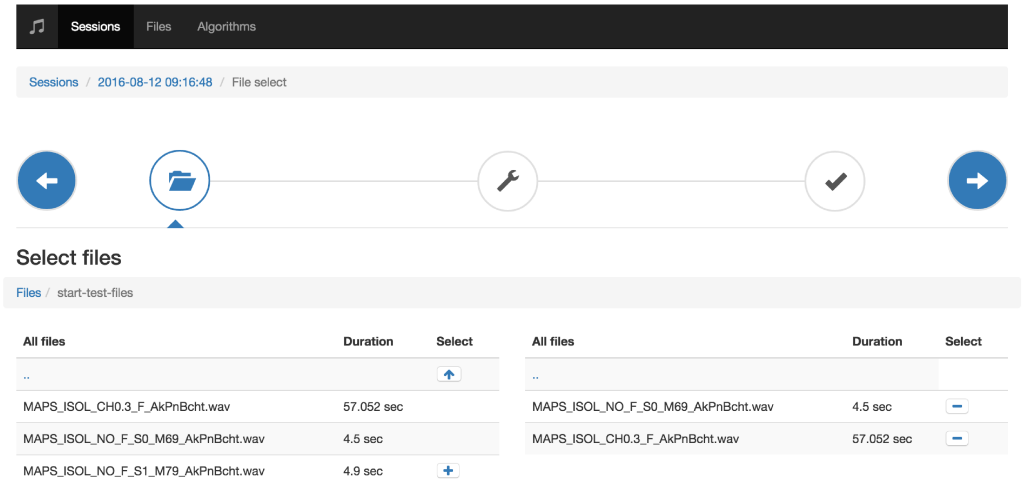
- izbira zvočnih datotek (Slika 4.6)
- izbira in konfiguracija algoritmov (Slika 4.7)
- pregled, zagon in nadzor izvajanja procesa (Slika 4.8)

V prvih dveh korakih določimo množico testnih datotek in nabor transkripcijskih algoritmov. Za vsak algoritem imamo možnost konfiguracije vhodnih parametrov, ki smo jih definirali v procesu dodajanja novega algoritma. Zadnji korak je namenjem pregledu nastavitev in zagonu procesa. Tu imamo dodatno možnost vpogleda na izhod sistemske konzole. S tem lahko vidimo, če smo v procesu konfiguracije algoritma katero od nastavitev napačno definirali ter zajamemo informacije sistemske konzole v primeru, da ta določenih akcij ne podpira.

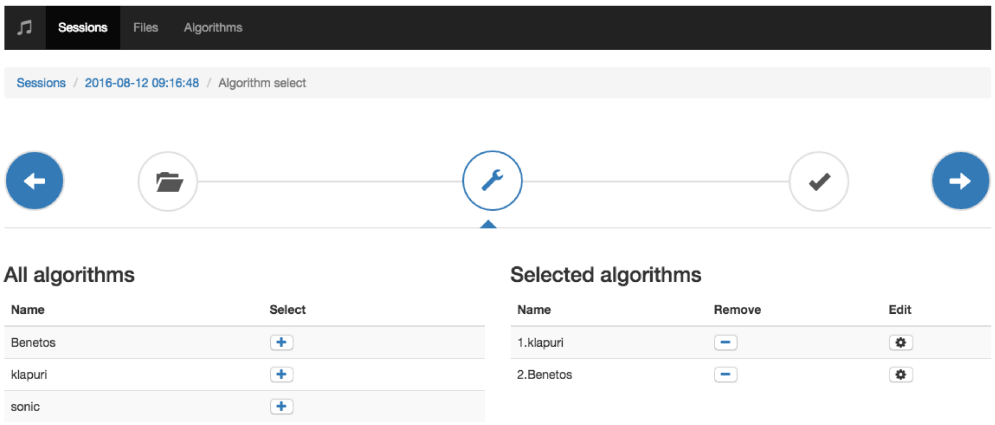
Po končanem procesu transkripcije nam sistem omogoča natančen pregled rezultatov (Slika 4.9). Za to poskrbi poseben del aplikacije. V tem delu najprej izberemo željeno testno datoteko in algoritem, ki smo ga definirali v prejšnjih korakih. Preko elementa za vizualizacijo transkripcije lahko nato ovrednotimo točnost rezultatov. Ker nam grafični vmesnik dovoljuje sočasno vizualizacijo poljubnega števila transkripcij, lahko na ta način primerjamo rezultate več algoritmov hkrati. S pomočjo nastavitve prosojnosti in barvne sheme rezultata lahko lažje razlikujemo med hkratnimi zapisi klavirskih tablatur. Na grafični vmesnik lahko dodamo tudi referenčno transkripcijo, v kolikor je ta prisotna v zbirki testnih datotek. Grafični pregled transkripcije smo rešili s pomočjo notacije klavirske tablature (piano-roll). Ker hočemo rezultate transkripcije oceniti tudi s pomočjo drugih kriterijev, smo v sistem vgradili pregled nad evalvacijskimi merami MIREX (Slika 4.10). Te lahko primerjamo na nivoju posameznega algoritma ali pa preko vseh izbranih algoritmov in datotek.

Rezultate transkripcije lahko dodatno obdelujemo preko orodij, ki so uporabniku na voljo v orodni vrstici. Tako lahko tone na klavirski tablaturi dodajamo in odstranjujemo. Pri tem si lahko pomagamo s povečavo in premikanjem vzdolž grafičnega prikaza klavirske tablature. Poleg tega lahko za vsak izbrani algoritem določimo tudi prag propustnosti magnitude aktivacij. S pomočjo tega praga nastavimo mejo, pod katero aktivacije ne bodo več vključene v vizualizacijo. Za pregled nad izbrano zvočno datoteko je na voljo vgrajen predvajalnik glasbe. Prav tako lahko rezultat transkripcije preverimo z vgrajenim predvajalnikom MIDI. Za pregled nad valovno obliko zvočnega posnetka poskrbi dodatni grafični element. Ta nam omogoča vpogled v glasnost in trenutno pozicijo predvajanja. Če smo z rezultati transkripcije zadovoljni, lahko klavirske tablature izvozimo v zapisu MIDI

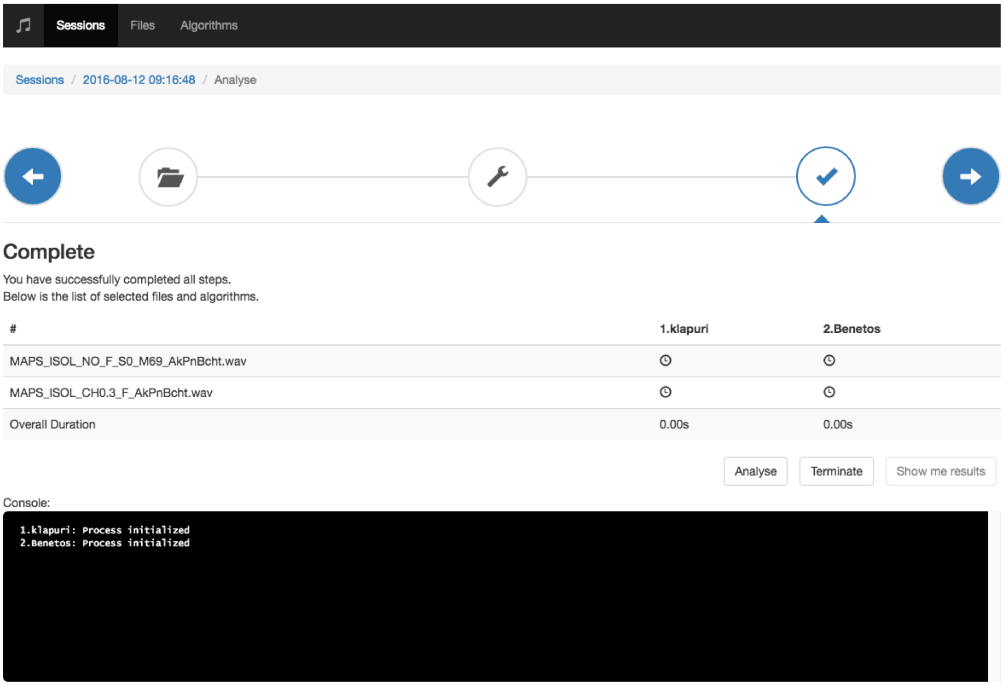
in uporabimo v drugih orodjih, kot so na primer sintetizatorji zvoka.



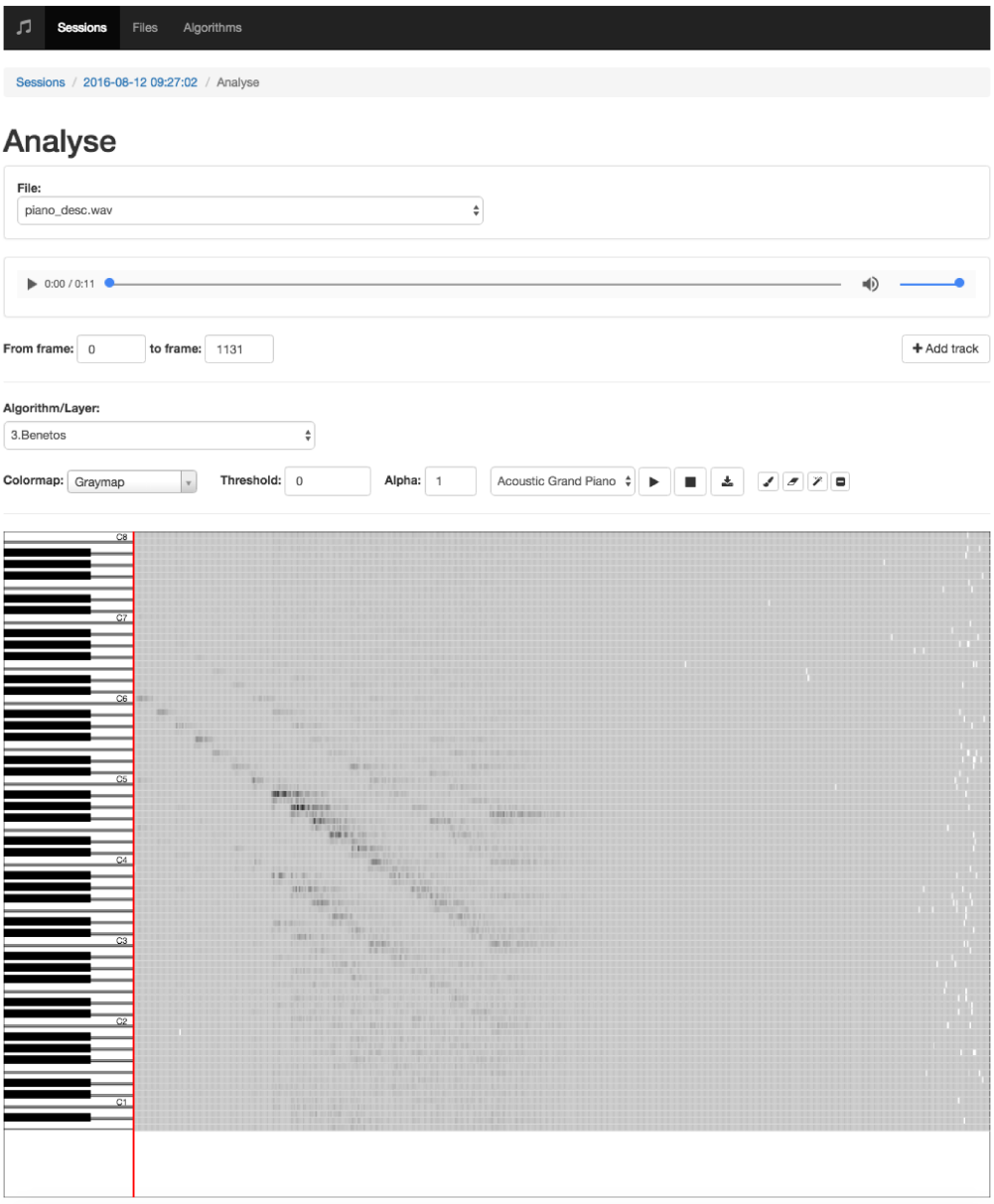
Slika 4.6: Grafični vmesnik za izbiro datotek, nad katerimi izvajamo analizo



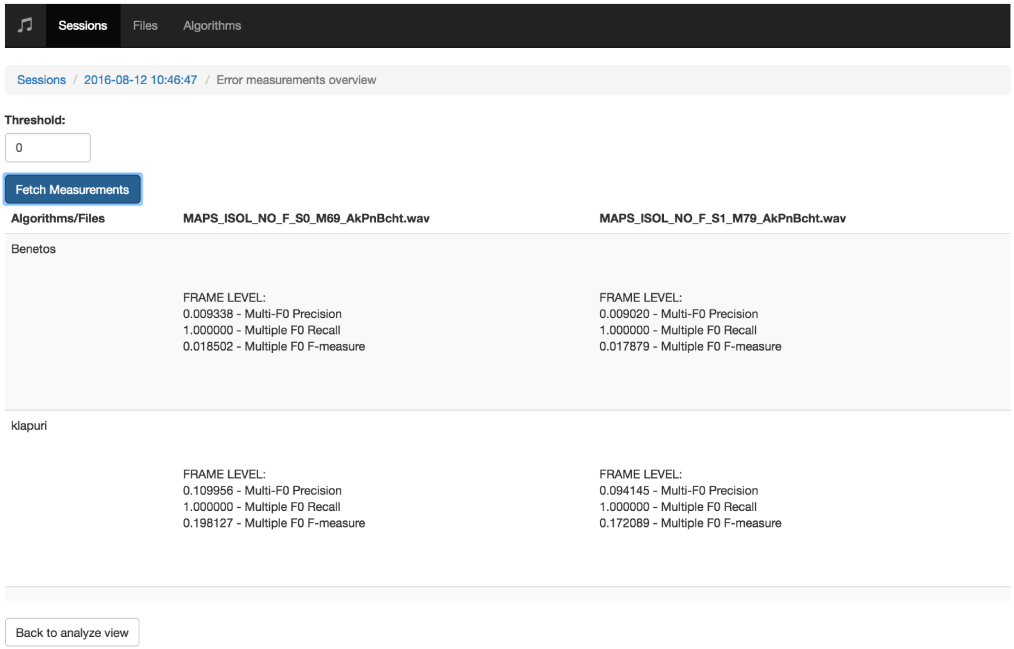
Slika 4.7: Grafični vmesnik za izbiro transkripcijskih algoritmov



Slika 4.8: Grafični vmesnik za pregled nad konfiguracijo analize ter kontrola nad izvajanjem



Slika 4.9: Grafični vmesnik za pregled nad rezultati transkripcije



Slika 4.10: Grafični vmesnik za pregled evalvacijskih mer nad izbranimi algoritmi in datotekami

4.4 Implementacija delov sistema

V tem delu sta opisana dva pomembnejša sklopa implementacije orodja. Postopek zagona in nadzora algoritmov ter normalizacija rezultatov transkripcij na strani strežnika.

Zagon analize in nadzor izvajanja algoritmov

Vsak proces transkripcije (seja) se v sistemu odraža kot nova instanca razreda *Session*. Ta poleg lastnosti, kot so ime seje, seznam izbranih datotek in datuma zadnje spremembe, vsebuje tudi seznam izbranih algoritmov. Ko v sistemu tak proces poženemo, aplikacija za vsak algoritem ustvari novo instanco razreda, poimenovano $AlgorithmProcess$, ki zajema vse nastavitve, določene s strani uporabnika. S pomočjo podatkov, ki jih je uporabnik definiral v procesu dodajanja algoritma, upravitelj procesov ($ProcessManager$) sestavi ukaz in parametre ter

izbere izvajalno okolje (konzola Octave ali sistemska konzola okolja Windows). Vsaka instanca zase poskrbi za zajem rezultatov algoritma ali morebitnih napak.

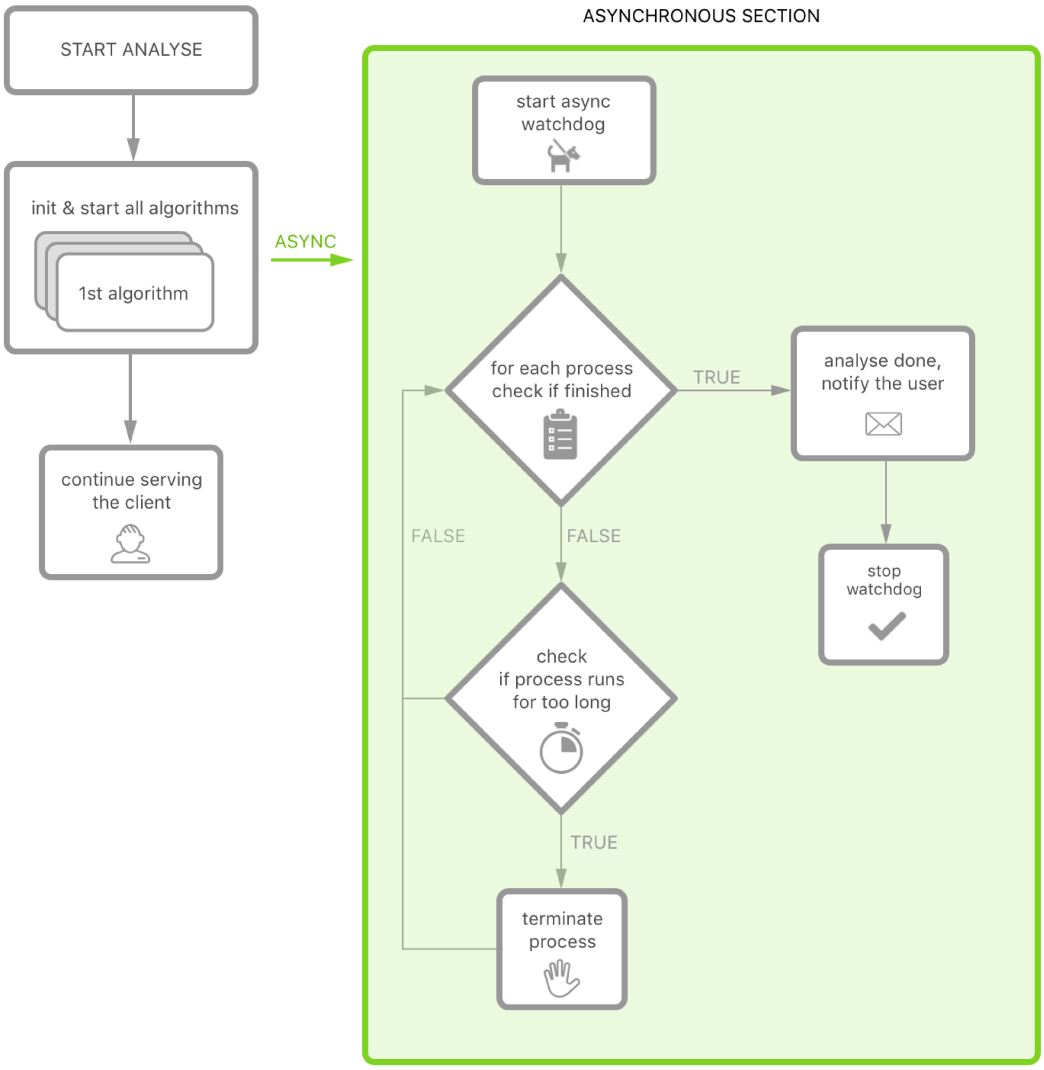
Da lahko nad izvajanjem algoritmov asinhrono vršimo nadzor, smo morali vsak proces definirati na način brez oviranja (non-blocking mode). Na ta način preprečimo morebitne zaklepe sistema v primeru napak med izvajanjem algoritmov. To bi se na primer lahko zgodilo, ko algoritem nikoli ne vrne izhodnega statusa.

Izvajanje algoritmov asinhrono nadzira metoda, ki je definirana na nivoju seje. Za implementacijo asinhrone funkcionalnosti smo uporabili ukaza async/await ogrodja .NET, ki s pomočjo prevajalnika v razvojnem okolju Visual Basic 2012 (ali novejše) programerju to poenostavi. Ta metoda v določenih časovnih intervalih nadzira izvajanje vseh procesov znotraj trenutne seje. Z asinhronim pristopom omogočimo nemoteno izvajanje preostale logike v trenutni seji (npr. serviranje trenutnega stanja analize odjemalcu v določenih intervalih).

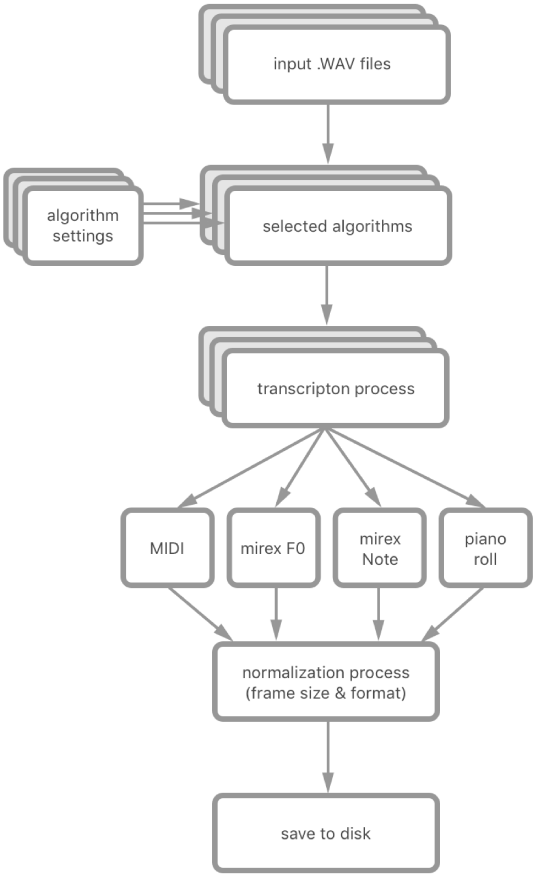
Da izboljšamo robustnost sistema, se med procesom nadzora preverja tudi čas izvajanja posameznega algoritma. Če je izvajanje enega izmed njih doseglo prag, sistem ta proces prisilno ustavi. Prag je določen med postopkom izbire algoritmov s strani uporabnika. Privzeta vrednost praga je 10 minut. Pregled nad delovanjem sistema orisuje (Slika 4.11).

Normalizacija rezultatov transkripcije

Da lahko zagotovimo hkratno vizualizacijo algoritmov in evalvacijo natančnosti, moramo poskrbeti, da so rezultati vseh transkripcij v isti obliki. Transkripcijski algoritmi namreč zvočni posnetek razstavijo na manjše časovne odseke, ki pa so lahko različnih dolžin. Hkraten pregled rezultatov v taki obliki je nejasen ter neprimeren za evalvacijo. Sistem zato po vsakem zaključku izvajanja transkripcije nad rezultati izvrši normalizacijo, ki poenoti dolžino vseh časovnih odsekov na 10ms, ter vrednosti zapiše v datoteko v formatu klavirske tablature (Slika 4.12). Ta datoteka je nato osnova za prikaz transkripcije in izračun evalvacijskih mer.



Slika 4.11: Grafični prikaz proženja in nadzora izvajanja procesov na strani strežnika



Slika 4.12: Grafični prikaz procesa normalizacije različnih formatov na izhodu algoritmov

Poglavje 5

Uporaba orodja v praksi

V tem poglavju bomo predstavili delovanje orodja in evalvacijo na primeru dveh vhodnih zvočnih posnetkov in treh algoritmov.

5.1 Konfiguracija analize

5.1.1 Vhodni zvočni posnetki

Za osnovo smo izbrali dve datoteki iz zbirke MAPS.

- MAPS_UCHO_C0-4-9_I60-68_S1_n7_AkPnBcht.wav (Dat 1)
- MAPS_MUS-chpn-p4_AkPnBcht.wav (Dat 2)

Za izbiro smo se odločili na podlagi različne stopnje polifonije in dolžine posnetka. Prva datoteka (Dat 1) vsebuje zvočni zapis primera akorda, ki se pogosto pojavi v kompozicijah zahodne klasične in jazz glasbe. Stopnja polifonije je 3, dolžina zvočnega zapisa pa je 3.9 sekunde. Druga datoteka (Dat 2) vsebuje zvočni zapis kompozicije Frédérica Chopina, "$PreludeinE - Minor(no.4)$". Dolžina zapisa je 1.44 minute. Oba posnetka sta zaigrana na simulator inštrumenta Bechstein D 280 v akustičnih pogojih koncertne dvorane s pomočjo programa Native Instruments.

5.1.2 Testni algoritmi in izbira vhodnih parametrov

Transkripcijo smo izvedli na treh algoritmih. Vrednosti posameznih vhodnih parametrov so navedene v nadeljevanju.

Klapuri

Sampling rate of input (fs): 44100
Size of analysis frame (frameSz): 1024
Frame hop used when analysing input (frameHop): 512
Number of F0s to extract in each frame (Npoly): 10
LagPars vector [lagMin, lagMax, lagPrec]: [20, 4200, 0.1]
HsumPars vector: [0.1, 27, 1.4]

Benetos

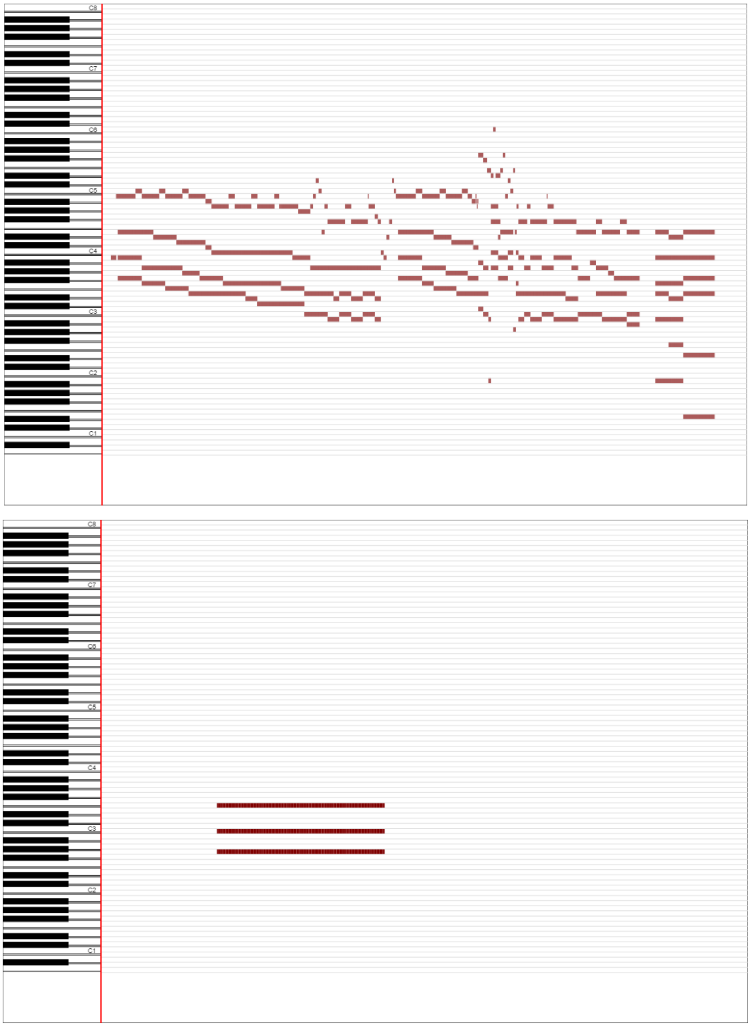
Number of itrerations: 1
Number of sources: 3
Sparsity parameter for pitch activation: 1
Sparsity parameter for source contribution: 1
Sparsity parameter for pitch shifting: 1

SONIC

NetworkDir: privzeta pot do direktorija mrež
Tuning: 10
Strattime: 0
Endtime:0
Maxthreads: -1
Onsets: privzeta prazna vrednost
Output type: zapis MIDI

5.2 Rezultati in evalvacijske mere glede na referenčno anotacijo

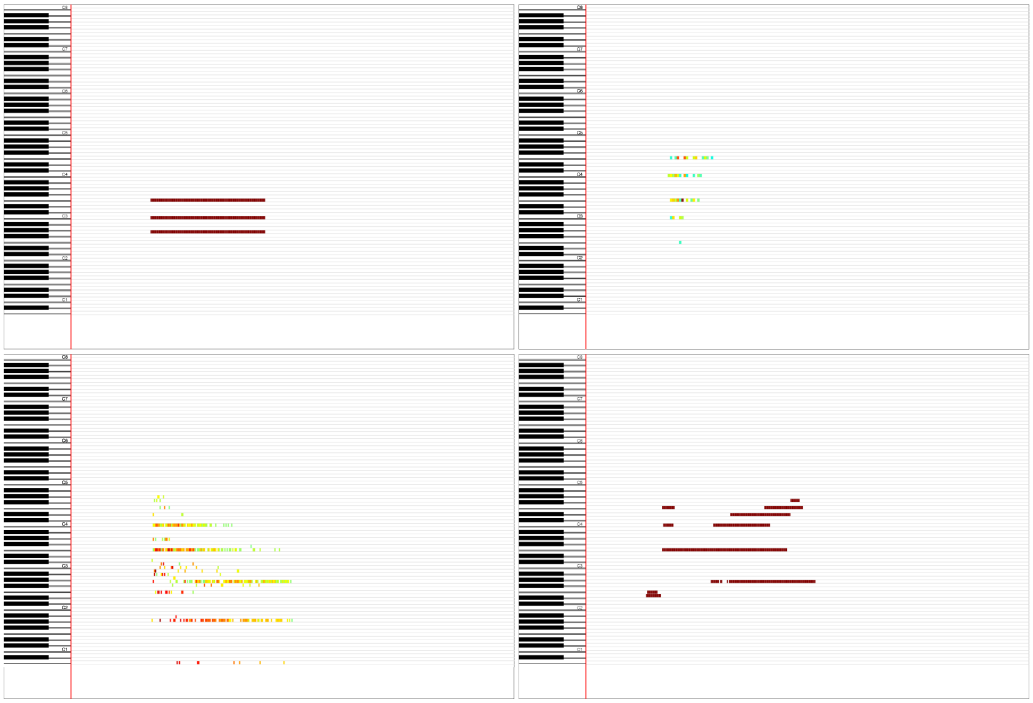
V nadeljevanju so prikazani rezultati transkripcij ter evalvacijskih mer v primerjavi z referenčnim rezultatom (Slika 5.1).



Slika 5.1: Zgoraj, prikaz referenčne anotacije datoteke (Dat 2), spodaj, prikaz referenčne anotacije datoteke (Dat 1)

5.2.1 Transkripcija datoteke (Dat 1)

Transkripcija je bila izvedena nad datoteko (Dat 1) ter algoritmi Benetos, Klapuri in SONIC. Klavirske tablature so prikazane na sliki (Slika 5.2). V nadeljevanju so navedeni rezultati evalvacijskih mer za posamezen algoritem.



Slika 5.2: Klavirske tablature pri transkripciji datoteke (Dat 1). Zgoraj levo je referenčna klavirska tablatura, zgoraj desno rezultat Benetos, spodaj levo rezultat Klapuri, spodaj desno rezultat SONIC

Benetos

Vrednosti evalvacijskih mer pri pragu magnitude aktivacij 0.4:

Multi-F0 Precision: 0.378378
Multiple F0 Recall: 0.092409
Multiple F0 F-measure: 0.148541
Predominant F0 Accuracy: 0.000000

Multiple F0 Accuracy: 0.080229
Multiple F0 lower octave error rate: 0.000000
Multiple F0 upper octave error rate: 0.080229
Multiple F0 Precision without counting octave errors: 0.756757
Multiple F0 Recall without counting octave errors: 0.184818
Multiple F0 F-measure without counting octave errors: 0.297082
Multiple F0 Accuracy without counting octave errors: 0.174455
Multiple F0 Error total: 0.920792
Multiple F0 Error substitution: 0.138614
Multiple F0 Error miss: 0.768977
Multiple F0 Error false alarm: 0.013201

Klapuri

Vrednosti evalvacijskih mer pri pragu magnitude aktivacij 0.5:

Multi-F0 Precision: 0.434641
Multiple F0 Recall: 0.438944
Multiple F0 F-measure: 0.436782
Predominant F0 Accuracy: 0.000000
Multiple F0 Accuracy: 0.279412
Multiple F0 lower octave error rate: 0.000000
Multiple F0 upper octave error rate: 0.121849
Multiple F0 Precision without counting octave errors: 0.624183
Multiple F0 Recall without counting octave errors: 0.630363
Multiple F0 F-measure without counting octave errors: 0.627258
Multiple F0 Accuracy without counting octave errors: 0.456938
Multiple F0 Error total: 0.709571
Multiple F0 Error substitution: 0.422442
Multiple F0 Error miss: 0.138614
Multiple F0 Error false alarm: 0.148515

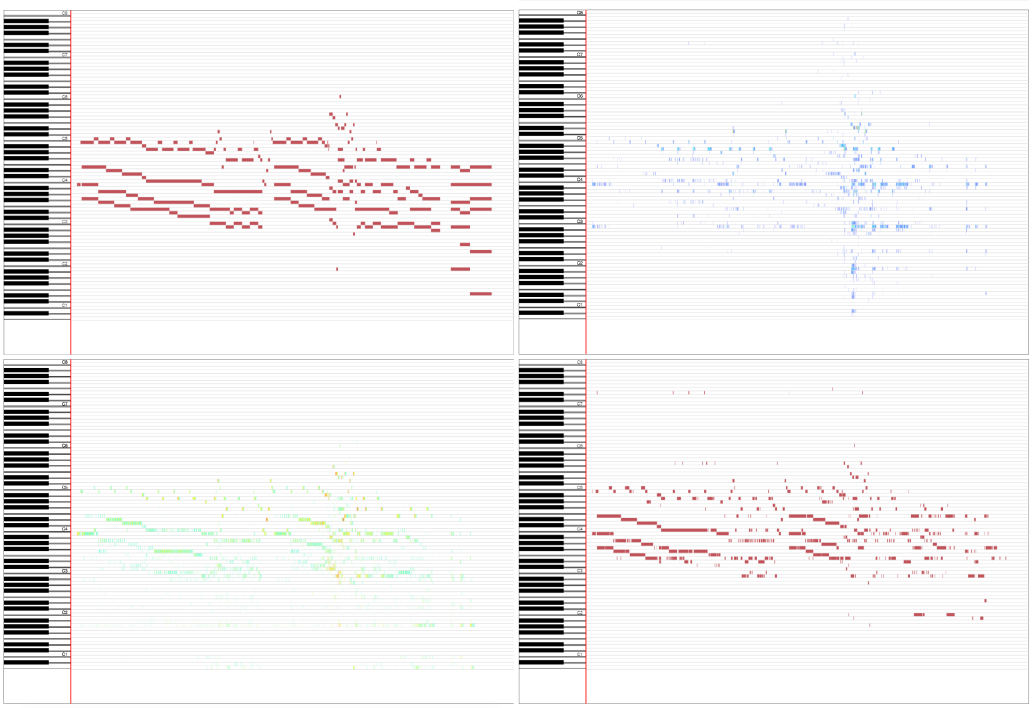
SONIC

Vrednosti evalvacijskih mer:

Multi-F0 Precision: 0.066667
Multi-F0 Recall: 0.333333
Multi-F0 F-measure: 0.111111
Accuracy: 0.058824
Average overlap ratio: 0.909087
Precision without considering octave errors: 0.133333
Recall without considering octave errors: 0.666667
F-measure without considering octave errors: 0.222222
Accuracy without considering octave errors: 0.117647
Average overlap ratio without considering octave errors: 0.494146

5.2.2 Transkripcija datoteke (Dat 2)

Transkripcija je bila izvedena nad datoteko (Dat 2) ter algoritmi Benetos, Klapuri in SONIC. Klavirske tablature so prikazane na sliki (Slika 5.3). V nadeljevanju so navedeni rezultati evalvacijskih mer za posamezen algoritem.



Slika 5.3: Klavirske tablature pri transkripciji datoteke (Dat 2). Zgoraj levo je referenčna klavirska tablatura, zgoraj desno rezultat Benetos, spodaj levo rezultat Klapuri, spodaj desno rezultat SONIC

Benetos

Vrednosti evalvacijskih mer pri pragu magnitude aktivacij 0.14:

Multi-F0 Precision: 0.596651
Multiple F0 Recall: 0.104845
Multiple F0 F-measure: 0.178350
Predominant F0 Accuracy: 0.000000

Multiple F0 Accuracy:0.097905
Multiple F0 lower octave error rate: 0.013980
Multiple F0 upper octave error rate: 0.008243
Multiple F0 Precision without counting octave errors: 0.732080
Multiple F0 Recall without counting octave errors: 0.128643
Multiple F0 F-measure without counting octave errors: 0.218832
Multiple F0 Accuracy without counting octave errors: 0.122858
Multiple F0 Error total: 0.917404
Multiple F0 Error substitution: 0.048628
Multiple F0 Error miss: 0.846527
Multiple F0 Error false alarm: 0.022249

Klapuri

Vrednosti evalvacijskih mer pri pragu magnitude aktivacij 0.4:

Multi-F0 Precision: 0.696572
Multiple F0 Recall: 0.279560
Multiple F0 F-measure: 0.398991
Predominant F0 Accuracy: 0.000000
Multiple F0 Accuracy: 0.249212
Multiple F0 lower octave error rate: 0.025471
Multiple F0 upper octave error rate: 0.004372
Multiple F0 Precision without counting octave errors: 0.779986
Multiple F0 Recall without counting octave errors: 0.313037
Multiple F0 F-measure without counting octave errors: 0.446769
Multiple F0 Accuracy without counting octave errors: 0.287639
Multiple F0 Error total: 0.723589
Multiple F0 Error substitution: 0.118628
Multiple F0 Error miss: 0.601812
Multiple F0 Error false alarm: 0.003149

SONIC

Vrednosti evalvacijskih mer:

Multi-F0 Precision: 0.142169
Multi-F0 Recall: 0.097682
Multi-F0 F-measure: 0.115800
Accuracy: 0.061458
Average overlap ratio: 0.451060
Precision without considering octave errors: 0.178313
Recall without considering octave errors: 0.122517
F-measure without considering octave errors: 0.145240
Accuracy without considering octave errors: 0.077083
Average overlap ratio without considering octave errors: 0.440596

5.3 Primerjava evalvacijskih mer in časovne zahtevnosti

Na sliki (Slika 5.4) so prikazani rezultati evalvacijskih mer za vse algoritme in datoteke, ki smo jih vključili v analizo.

Algorithms/Files	MAPS_MUS-chpn-p4_AkPnBcht.wav	MAPS_UCHO_C0-4-9_I60-68_S1_n7_AkPnBcht.wav
Benetos	FRAME LEVEL: 0.045085 - Multi-F0 Precision 0.999948 - Multiple F0 Recall 0.086280 - Multiple F0 F-measure	FRAME LEVEL: 0.021992 - Multi-F0 Precision 1.000000 - Multiple F0 Recall 0.043037 - Multiple F0 F-measure
klapuri	FRAME LEVEL: 0.269718 - Multi-F0 Precision 0.647885 - Multiple F0 Recall 0.380875 - Multiple F0 F-measure	FRAME LEVEL: 0.132266 - Multi-F0 Precision 0.518152 - Multiple F0 Recall 0.210738 - Multiple F0 F-measure
sonic	NOTE LEVEL: 0.157005 - Multi-F0 Precision 0.107616 - Multi-F0 Recall 0.127701 - Multi-F0 F-measure	NOTE LEVEL: 0.076923 - Multi-F0 Precision 0.333333 - Multi-F0 Recall 0.125000 - Multi-F0 F-measure

Slika 5.4: Rezultat analize izbranih datotek z izbranimi algoritmi

#	1.Benetos	2.klapuri	3.sonic
MAPS_MUS-chpn-p4_AkPnBcht.wav	✔ 726.54s	✔ 53.91s	✔ 282.42s
MAPS_UCHO_C0-4-9_I60-68_S1_n7_AkPnBcht.wav	✔ 19.97s	✔ 2.05s	✔ 23.52s
Overall Duration	746.51s	55.96s	305.93s

Slika 5.5: Časovna zahtevnost procesa transkripcije nad izbranimi datotekami z izbranimi algoritmi

5.3.1 Ugotovitve

Na podlagi dobljenih rezultatov (Slika 5.5) vidimo, da je časovno najmanj zahteven Klapurijev algoritem, največ časa pa potrebuje Benetos. Najmanjše število

zgrešenih detekcij osnovnih frekvenc (Multi F0 precision) zagotovi Klapuri. Procent pravilno detektiranih F0 (Muliple F0 recall) je najboljši v primeru algoritma Benetos. Uporaba algoritma SONIC prikazuje podporo primerjave sistemov, ki pristopajo k implementaciji na osnovi sledenja tonov (note-level). Poleg tega lahko vidimo, da je klavirska tablatura v primeru uporabe algorima SONIC zelo blizu referenčnemu rezultatu.

Poglavje 6

Zaključek

V sklopu diplomskega dela smo razvili enostavno in uporabno orodje za primerjavo transkripcijskih algoritmov. Aplikacija uporabniku omogoča pregled vseh pomembnejših primerjalnih kriterijev. To so, časovna zahtevnost izvajanja, grafični prikaz in primerjava rezultatov z ostalimi algoritmi ter pregled vrednosti evalvacijskih mer MIREX. Poleg tega nam sistem omogoča tudi dodajanje lastnih algoritmov. S pomočjo sodobnih tehnologij za razvoj spletnih aplikacij smo omogočili delovanje sistema v vseh širše uporabljenih brskalnikih. Na primeru nekaj testnih algoritmov smo prikazali delovanje aplikacije v praksi. Orodje smo avgusta 2016 predstavili navzočim na mednarodni konferenci pridobivanja informacij iz glasbe v New Yorku.

6.1 Prednost in omejitve

Aplikacija na enem mestu združuje pregled glavnih kriterijev za vrednotenje delovanja transkripcijskih sistemov. Na ta način lahko uporabnik hitro določi prednosti in slabosti tovrstnih algoritmov. Prednost aplikacije je tudi možnost dodajana lastnih rešitev. S tem razvijalcu pomagamo, da hitro oceni konkurenčnost svojega sistema z že obstoječimi. Enostavno uporabo orodja omogoča intuitiven grafični vmesnik, s pomočjo katerega lahko transkripcije predvajamo, dodatno obdelujemo in izvozimo v zapis MIDI. Z vgrajenim urejevalnikom datotek lahko razširimo tudi zbirko zvočnih datotek. Obseg te množice je ključnega pomena za pravilno evalvacijo algoritmov. Ker je proces transkripcije časovno zahtevna operacija, upo-

rabniku omogočimo obveščanje o zaključku preko sporočila v obliki elektronske pošte.

Aplikacija je trenutno omejena na dodane algoritme, ki so implementirani v obliki skript Matlab in izvršljive datoteke Windows okolja. Ker pa je implementacija modularna, je razširitev trivialna. Druga omejitev je format zapisa transkripcije na izhodu algoritma. Čeprav smo v orodje vgradili možnost branja vseh najbolj razširjenih formatov (klavirska tablatura, datoteka MIDI, MIREX F0, MIREX Note), mora uporabnik v primeru drugačnega zapisa transkripcije sam poskrbeti za preoblikovanje izhoda algoritma v eno od podprtih anotacij. To smo pokazali na primeru Klapurijevega algoritma, kjer smo izhod v obliki matrike osnovnih frekvenc F0 in njihovih navzočnosti preoblikovali v klavirsko tablaturo. Urejevalnik transkripcij zaenkrat ne podpira koncepta takta.

6.2 Nadaljni razvoj in izboljšave

Orodje želimo ponuditi vsem, ki raziskujejo na področju pridobivanja informacij iz glasbe. Da to dosežemo, je potrebno implementirati sistem za registracijo in prijavo uporabnikov, kar pomeni razširitev strežniškega dela s podatkovno bazo. Določene dele implementacije, kot je npr. normalizacija izhoda algoritma, je možno še dodatno optimizirati. Na strani odjemalca je možno prikaz rezultatov izboljšati z izbiro le željenih evalvacijskih mer. V sistem lahko v prihodnosti vključimo še nekaj dodatnih zbirk testnih datotek in transkripcijskih algoritmov.

Literatura

[1] James A. Moorer. On the Transcription of Musical Sound by Computer. Computer music journal, 1(4):32, 1977.

[2] Martin Piszczalski and Bernard A. Galler. Predicting musical pitch from component frequency ratios. Journal of the Acoustical Society of America, 66(3):710–720, September 1979.

[3] Maher, R.C. An approach for the separation of voices in composite music signals. Ph.D. thesis, University of Illinois, Urbana, pages 1-6, 1989

[4] Chafe, C., & Jaffe, D. Source separation and note identification in polyphonic music. In: Proc. IEEE International Conf. on Acoust., Speech, and Signal Processing, Tokyo, 1289–1292, 1986.

[5] K. Kashino, K. Nakadai, T. Kinoshita, and H. Tanaka. Organisation of hierarchical perceptual sounds: Music scene analysis with autonomous processing modules and a quantitative information integration mechanism. In International Joint Conference on Artificial Intelligence, pages 158–164, Montreal, Quebec, 1995

[6] M. Goto. A predominant-F0 estimation method for real-world musical audio signals: Map estimation for incorporating prior knowledge about f0s and tone models. In Proc. Workshop on Consistent and reliable acoustic cues for sound analysis, Aalborg, Denmark, page 1, 2001.

[7] M. Ryynänen and A. Klapuri. Polyphonic music transcription using note event modeling. In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, USA, page 1 2005

[8] K. D. Martin. Automatic transcription of simple polyphonic music: robust front end processing. Technical Report 399, MIT Media Laboratory Perceptual Computing Section, pages 1-2, 1996.

[9] Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. EURASIP Journal on Applied Signal Processing, 2007(1):154, 2007.

[10] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Discriminative non-negative matrix factorization for multiple pitch estimation. In ISMIR, pages 205–210, 2012.

[11] Matija Marolt. A connectionist approach to automatic transcription of polyphonic piano music. Multimedia, IEEE Transactions on, 6(3):439–449, 2004.

[12] Matevž Pesek, Aleš Leonardis, and Matija Marolt. A compositional hierarchical model for music information retrieval. In Proceedings of the International Conference on Music Information Retrieval (ISMIR), pages 131–136, Taipei, 2014.

[13] E. Benetos and T. Weyde, "An efficient temporally-constrained probabilistic model for multiple-instrument music transcription", in Proc. 16th International Society for Music Information Retrieval Conference, page 701, Oct. 2015.

[14] Matija Marolt. Sonic: Transcription of polyphonic piano music with neural networks. In Workshop on Current Research Directions in Computer Music, pages 217–224, 2001.

[15] Klapuri, A., "Multiple fundamental frequency estimation by summing harmonic amplitudes", 7th International Conference on Music Information Retrieval, Victoria, Canada, page 1, Oct. 2006.

[16] Gareth Loy. Musicians make a standard: the midi phenomenon. Computer Music Journal, pages 8-9, 1985.

[17] Toiviainen, P., & Eerola, T. (2016). MIDI Toolbox 1.1. URL: https://github.com/miditoolbox/1.1

[18] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle. IEEE Transactions on Audio, Speech, and Language Processing, 18(6):1643–1654, August 2010.

[19] Valentin Emiya, Nancy Bertin, Bertrand David, and Roland Badeau. Maps-a piano database for multipitch estimation and automatic transcription of music. 2010.

[20] Fielding, R. (ur). Architectural Style and the Design of Network-based Software Architecture. 29.8.2012

[21] MIDI. Specification. http://www.midi.org/, 2001.

[22] Music Information Retrieval Evaluation eXchange(MIREX). http://music-ir.org/mirexwiki/, 2011.

[23] M. Bay, A. F. Ehmann, and J. S. Downie. Evaluation of multiple-F0 estimation and tracking systems. In ISMIR, pages 315–320, 2009.

[24] E. Benetos and T. Weyde, "An efficient temporally-constrained probabilistic model for multiple-instrument music transcription", in Proc. 16th International Society for Music Information Retrieval Conference, Oct. 2015.

[25] Benetos, Emmanouil, Simon Dixon, Dimitri Giannoulis, Holger Kirchhoff, and Anssi P. Klapuri. Automatic Music Transcription: Breaking the Glass Ceiling. Paper read at International Society for Music Information Retrieval Conference (ISMIR), at Porto, Portugal. 2012

[26] E. Benetos, S. Dixon, D. Giannouis, H. Kirchhoff and A. Klapuri. Automatic music transcription: challenges and future directions. Journal of Intelligent Information Systems, 41(3), 407–434, 2013.

[27] Z. Duan, E. Benetos. Automatic music transcription. ISMIR. 2015. url: http://c4dm.eecs.qmul.ac.uk/ismir15-amt-tutorial/.

[28] Vogels, J., Music Technology Course MUMT621 Presentation, McGill University, 12 March 2013,

[29] Klapuri, A., Introduction to Music Transcription. Institute of Signal Processing, Tampere University of Technology, Korkeakoulunkatu 1, 33720 Tampere, Finland Dec. 2006

[30] M. Paleari, B. Huet, A. Schutz and D. Slock, "A multimodal approach to music transcription,"2008 15th IEEE International Conference on Image Processing, San Diego, CA, 2008, pp. 93-96.

[31] Martin, K.D. (1996a). A blackboard system for automatic transcription of simple polyphonic music. MIT Media Laboratory Perceptual Computing Section Technical Report No. 385.

[32] Bello, J.P., Monti, G., Sandler, M. Techniques for automatic music transcription, Department of Electronic Engineering, King's College London, Strand, London, pages 6-7, 2000

[33] G. E. Poliner, D. P. W. Ellis, and A. F. Ehmann: "Melody Transcription from Music Audio: Approach and Evaluation," IEEE Transactions on Audio, Speech, and Language Processing, Vol. 15, N0. 4, pp. page 7, 2007.

[34] (2016) Asynchronous Programming with async and await. Available at: https://msdn.microsoft.com/en-us/library/mt674882.aspx

[35] (2016), AngularJS documentation. Dostopno na: https://docs.angularjs.org/guide/

[36] (2016) Note names, MIDI numbers and frequencies. Available at: http://newt.phys.unsw.edu.au/jw/notes.html

[37] (2016) Matlab documentation. Available at: http://www.mathworks.com/help/matlab/

[38] (2016) Online piano player. Available at: http://piano-player.info/

[39] (2016) MIDI Note Number to Frequency Conversion Chart. Available at: http://subsynth.sourceforge.net/midinote2freq.html

[40] Downie, J. Stephen (2008). The Music Information Retrieval Evaluation Exchange (2005-2007): A window into music information retrieval research.Acoustical Science and Technology 29 (4): 247-255.

[41] Downie, J. Stephen, Andreas F. Ehmann, Mert Bay and M. Cameron Jones. (2010). The Music Information Retrieval Evaluation eXchange: Some Observations and Insights. Advances in Music Information Retrieval Vol. 274, pp. 93-115